A TIME-SHARING DEBUGGING SYSTEM FOR A SMALL COMPUTER

J. McCarthy
Computation Center, Stanford University, Stanford, California
S. Boilen
Bolt Beranek and Newman Inc., Cambridge, Mass.
E. Fredkin
Information International Inc., Maynard, Mass.
J. C. R. Licklider
Advanced Research Projects Agency, Department of Defense

The purpose of the BBN time-sharing system is to increase the effectiveness of the PDP-1 computer for those applications involving man-machine interaction by allowing each of the five users, each at his own typewriter to interact with the computer just as if he had a computer all to himself. The effectiveness of this interaction is further enhanced by the use of the TYC language for controlling the operation and modification of programs.

First the computer. The PDP-1* is a single address binary computer with an 18 bit word and five microsecond memory cycle; most instructions require ten microseconds to execute. The basic memory size is 4096 words, but up to 65,536 words may be addressed indirectly. The machine we used has 8192 words, 4096 of which are reserved for the time-sharing system. Each user sees a 4096 word memory. We shall describe further relevant features of the computer later.

Attached to the computer is a high speed magnetic drum memory divided into 22 fields each of 4096 words. A basic operation of the drum system is the *memory-swap* accomplished in 33 milliseconds. In this operation 4096 words are transferred from the core memory to a drum field and simultaneously the core memory is loaded from a different drum field. This permits the following time-sharing mode of operation.

A 4096 word drum field is allocated for saving the *core image* of each user when his program is not running. A user's program in *run status* is run for 140 milliseconds, then if there is another user also in run status, the state of core memory is stored in the first user's core image on drum and simultaneously the second user's core image is loaded into core and the second user's program is started in the appropriate place. In the worst case of all five users in run status, the system makes its rounds in $5 \times (140 + 33) = 865$ milliseconds. For man-machine interaction this means that if the user types a character calling for a response from his program that requires less than 140 milliseconds (= 14,000 machine instructions), he will get the first character of the response in less than .865 seconds. This worst case is ex-

* The PDP-1 computer is manufactured by Digital Equipment Corporation of Maynard, Massachusetts. All the equipment described in this paper was made by D.E.C. and their cooperation in developing and building the modifications and additions to the basic computer required for time-sharing was essential to the success of the project.

pected to be rare because when a user's program types out a multicharacter message, successive characters go into a buffer area in the system core; when the buffer is full the user's program is removed from run status until the buffer is nearly empty. Therefore, users with extensive output spend very little time in run status and the other users get correspondingly quicker service. (In fact the condition in which no users are in run status is expected to be so common that a provision for a background program to be run only when no time-sharer is in run status is contemplated.)

THE TYC CONTROL LANGUAGE

The language used to control the debugging is adapted from the DDT language devised by the TX-0 and PDP-1 group at M.I.T. directed by Jack Dennis for the TX-0 and PDP-1 computers. The use of typewriters rather than the console switches for on-line debugging has been developed at M.I.T. and M.I.T. Lincoln Laboratory since 1957, and at BBN since 1961. These languages have greatly increased the effectiveness of the TX-0, TX-2 and PDP-1 computers and are now being developed for the IBM 7090 time-sharing system by F. J. Corbato of the M.I.T. Computation Center. Unfortunately, except for a recent paper by Corbato, the work has not been published. Our only bow at history is to credit John Gilmore with the first such system for the TX-0 in 1957.

First, we shall consider the facilities for examining and changing registers. Suppose the user wants to examine register 344. He types

344/

and the computer types back the contents of this register interpreted as a computer instruction if possible. Thus it might type back "add 4072". The user then has several options.

1. He can carriage return and close the register if he is satisfied with its contents.
2. He can ask to see the next register by hitting the backspace key. The computer might then type back

345/ sub 4075.

3. He can ask to see the previous register.
4. He can ask to see register 4072 by hitting the tab key.
5. He can ask for the contents of the register as an octal or decimal number.
6. He can change the contents of the register by typing new contents such as "add 4073" and then hitting one of the delimiter characters.

The computer user who does not have experience with systems of this kind may not immediately realize the increase in effectiveness that these facilities alone give over console debugging. The advantages are that typing is easier than flipping switches, that a record is obtained of what was done, and that useful features can be added to the control language.

As a further feature the user can define symbolic names of addresses so that his typeouts can take the form

a/ add b + 3
a + 1/ sub b + 6

In order to start a program say at register 3145 the user types 3145G. This gives the typewriter up to the program, but he can get it back for control purposes by typing *center dot.* If he does so he can interrogate and change registers without stopping his program but if he wants to stop it he types H. Once it is stopped he can start it where it left off by typing G without a numerical argument. He can also interrogate the arithmetic registers either while the program is running or while it is stopped. Of course, the contents of a register interrogated when the program is interrupted at a random moment may not be significant; this depends on the program.

Except at the beginning and the end of his session the user does not ordinarily use the paper tape apparatus. Instead he designates a position on the drum for the punch and a position for the reader using TYC. An instruction in the user's program to punch a character onto paper tape actually results in entering the character into a buffer for transmission to the drum when the buffer is full or no punch instructions have been given for a while. This feature allowed us to make available in the time-sharing system symbolic assembly programs and other utility programs that were developed previously.

The TYC language is described in detail in reference 7.

RELEVANT FEATURES OF THE PDP-1 COMPUTER

In order to explain the detailed functioning of the BBN time-sharing system, it is necessary to understand the input-output system of the PDP-1 computer, the sequence break system and the restricted mode.

1. Typewriters and paper tape

Each *iot* instruction transfers a single character between the 18 bit *io* register of the computer and the external device. All *io* instructions have the same left 5 bits, the 6th bit is normally used to determine whether the wait before the next instruction is determined by the external device or whether the program keeps this responsibility. The remaining 12 bits of the instruction are used to determine the device and the direction of transfer. The characters are 8 bits on paper tape and 6 bits for typewriters. When the computer is not in sequence break mode a character typed by the user results in turning on a sense flag which can be interrogated by the program to determine when to pick up the character.

2. Drum

In transfers from the drum the number of words to be transferred and the locations in core and on the drum are specified. During the transfer the arithmetic unit of the machine is tied up.

3. The machine has other input-output devices but their operation does not have to be described in this paper.

4. The sequence-break system.

Besides the simple form of input-output described above, the machine has a sequence-break system which permits input-output and computation to be carried out simultaneously and asynchronously. The BBN time-sharing system makes full use of sequence break.

The sequence-break system has 16 channels to the outside world arranged in a priority chain with channel 0 having highest priority and channel 17_8 lowest priority. Associated with each channel are four registers in core 0. When a signal comes from an *io* device that the device has a character ready for the computer or is ready to receive a character from the computer, an interrupt will occur if an interrupt is presently allowed on that channel. When an interrupt occurs the contents of the *ac* (accumulator) *io* (input-output) and *pc* (program counter) registers are stored in three of the registers and control is transferred to the fourth which must therefore contain a jump to a program for dealing with the interrupt. When this program has finished its work it must execute an indirect jump through the register where the program counter was stored when the break started. This returns control to the program that was interrupted and tells the sequence break system that the break is over. Once a break is started on a channel a break cannot occur on a lower priority channel nor can another break occur on the same channel until the break is over. Each channel can store one break until it is allowed to occur.

5. Restricted Mode

This mode was devised specifically for the time-sharing system. When the computer is in this mode and if there is no break started in the sequence break system, then any of the following events lead to a sequence break on channel 16.

1. An attempt to obey an *io* instruction.
2. An instruction that would normally stop the computer such as a halt instruction or an illegal instruction.
3. An attempt to refer to core 0.

The instructions that enter and leave extend mode and restricted mode are considered *io* instructions. The time-sharing system operates only when a sequence break has occurred and hence is not subject to the restrictions.

6. The Channel 17 Clock.

Every 20 milliseconds a signal for a sequence break on channel 17 occurs. Programs can turn off channel 17 (or any other channel) by an *io* instruction if they don't want breaks to occur. Note that channel 17 is the lowest priority channel. The clock is a multivibrator whose period is controlled by a potentiometer.

HOW THE TIME-SHARING WORKS

The time-sharing executive program is not readily described by a single flow chart because its different parts act asynchronously as determined by sequence breaks. It includes the following programs:

1. The typewriter *io* program

Associated with each typewriter is an input-output program and a buffer area. These pro-

grams are entered when sequence breaks occur. Suppose a user types a character w. The program knows whether the character is addressed to the user's program or to TYC.

On the other hand, if the interrupt comes from the completion of the type-out of a character the program types the next character from the buffer if any.

After transferring the information the break is ended.

Channels 6, 7, 11, 14 and 15 are allocated to typewriters.

2. The channel 16 dispatcher.

The computer is in restricted mode during the operation of the time-sharing system. As we stated earlier, this means that *io* instructions and instructions that halt the machine lead to sequence breaks on channel 16. The user programs his input-output just as if there were no time-sharing system. Therefore, when a channel 16 break occurs the program first looks at the instruction that caused the break. Suppose the instruction is a type-out instruction. If the type-out buffer is not full the character that the user program wants to type is added to the buffer and if necessary a sequence break on the typewriter channel is instigated to start typing. If the type-out buffer is full the program must be dismissed from run status. If the instruction is a type-in instruction, a character is given to the user program if there is one in the buffer; otherwise the program must be dismissed from run status.

If the instruction is discovered to be one that halts the machine, the program is dismissed from run status and a note is left for TYC to tell the user what happened.

Paper tape input and output is handled in a similar way except that the dispatcher must check whether the user has the punch or reader assigned to him. If not, the user's program is dismissed, and a complaint is made to him. As soon as the reader or punch has been relinquished he can continue the program from where it left off.

The typewriter and paper tape instructions are interpreted and simulated by the channel 16 dispatcher so precisely that programs written before the time-sharing system was developed can be run without change in the system, provided they do not themselves use the sequence break system. This means that almost all the previously used symbolic assemblers, typewriter input-output routines, text editors etc. can be used without change and that the user can use the TYC language to debug routines that are to be used outside it.

3. The channel 17 clock routine.

Every 20 milliseconds or so a sequence break signal is given on channel 17. Since channel 17 is the lowest priority channel this break can take effect only when no typewriter, paper tape or channel 16 dispatcher break is in progress. Moreover, except when the channel 17 program turns off the sequence break system it can be interrupted by typewriter or paper tape sequence breaks.

The basic task of the channel 17 clock routine is to decide whether to remove the current user from core and if so to decide which user program to swap in as he goes out. A user may be removed from core for any of several reasons.

1. His quantum of time is up and he should be put on the tail of the queue.
2. He has filled an output buffer.
3. He has asked for a character and there is none in the input buffer.
4. He has tried to execute an illegal instruction or to use input-output equipment not available to him.
5. The typewriter control program has filled a buffer or has finished a request concerning his program.

If the channel 17 routine decides to remove the current user it will swap into core the next user in the round robin who is in run status. A user not in run status can become so for any of the following reasons.

1. An output buffer is almost empty.
2. A character requested by his program has arrived.
3. The typewriter control program wants him in core to interrogate registers, to change them, or to run the program.
4. The typewriter control program (TYC).

The typewriter control program is in core 0 and it interprets and obeys requests from the user to give him information about his program, to change it, to run it, and to stop it. The same program must work for all users and whenever a user is put in core TYC is modified so that it refers to the current user's program.

The user makes his requests and receives information from TYC using the same typewriter as his program uses for input and output. Therefore, it is important that no matter what program the user is attempting to debug he shall be able to regain control if it goes astray. This is accomplished as follows: When the user starts a program running he can either retain the typewriter for control purposes or else give it up to the program. If he gives the typewriter to the program, then characters it types appear on his typewriter and characters he types are given to his program if it asks for them. Suppose his bad program is taking characters from the typewriter but ignoring them. He can then type the character center dot "·" which is a non-spacing character on the PDP-1. If he follows this by a carriage return the typewriter is then in the control of TYC and subsequent characters are interpreted by TYC. If he actually wants *center dot* to be transmitted to his program, he must type it twice.

Suppose now that his program is in an output loop and refuses to stop typing. Then he turns the power off on the electric typewriter. The result of this is that the computer fails to get a "done pulse" from the next character typed within a second. Control then goes to TYC which tells the channel 17 program to dismiss the user's program from core and returns the typewriter to the control of TYC as soon as the power switch is turned on again.

APPLICATIONS

The most obvious application of the BBN Time-Sharing system is to speed up debugging by allowing each user more console time and good debugging languages. In our opinion the reduction in debugging time made possible by good typewriter debugging languages and adequate access to the machine is comparable to that provided by the use of ALGOL type languages for numerical calculation. Naturally, one would like to have both but this has not yet been accomplished on any machine.

We can now mention some other applications that our system makes possible by providing inexpensive console time.

1. *Small calculations.* At present there is a large gulf between desk calculators and computers. One can start getting results 10 seconds after sitting down at a desk calculator, but extensive calculations are very tedious. The BBN Time-Sharing System makes possible and economically reasonable providing a continuous transition from using the computer as a desk calculator at one extreme to writing ALGOL programs at the other. An intermediate step is a system that allows the user to define functions by statements like

$$f(x) = x \uparrow 2 + 3.0x \times + 4.3$$

or even

$g(m,n)$ = if $m>n$ then $g(n,m)$ else if rem $(n,m) = 0$ then m else $g(rem\ (n,m), m)$

and then be able to use these functions in arithmetic calculations by writing something like

$$g(3,21) \times f(38) + 19 =$$

and have the computer print the answer by interpreting the formulas for the functions. To some extent this has been achieved by the program "Expensive Desk Calculator" written by Robert Wagner at M.I.T.

2. *Editing Texts.* Several programs have been written to use the PDP-1 computer to edit paper tape texts. The user can originate text, make insertions and deletions, display the corrected text to make sure it is correct, find all occurrences of certain strings of symbols. These editing programs are much more convenient for correcting programs than using flexowriters or than making changes in a card

One such program, Colossal Typewriter, operates as follows:

There are two modes, text mode and control mode. When the program is in text mode, each character typed by the user is added to a buffer held in the core memory of the computer. There are four exceptions to this: If the user types a backspace, the program deletes the last character from the buffer, and this operation can be repeated as many times as may be required to correct a local error. Another character causes the program to type out the last 20 characters in the buffer, and a third returns the computer to control mode. The fourth special character—the single quote ' causes the cliche whose name is the following character to be entered in the text if there is such a cliche. Thus typing ' a causes the cliche named a to be entered. In addition, the cliche Feature may be used to enter any of the four control characters into the text. The user need only type ' followed by the character in question.

In control mode the user has the following facilities: to type out the buffer, to punch (pseudo-punch) the buffer, to read (pseudo-read from the drum) into the buffer a number of lines or until the first occurrence of a given cliche, to reset the ends of the buffer, to give the current buffer a name as a cliche, to kill the buffer, and to go into text mode.

Other text-editing programs allow the use of the CRT on the computer to display lines of text.

3. *Teaching Programs.* An experimental teaching laboratory using a system based on the principles of this paper is being installed at Stanford University.

Additional applications of large time-sharing systems are described in (2), (3) and (5).

OPERATING EXPERIENCE

The BBN Time-Sharing System has been in operation at Bolt Beranek and Newman Inc. since September 1962. The computer is operated in the time-sharing mode four hours per day. Initially, the number of typewriters was two but this has been increased to five. The present system has been found to have the following weaknesses which we hope will be corrected: There is no program library on the drum so that excessive use of the paper tape reader is required. Magnetic tape files for user programs are not available in the system. Five computer operated typewriters in one room make too much noise. Versions of the utility programs especially adapted to time-sharing are desired.

EXTENSION TO LARGER COMPUTERS

It is worthwhile to ask to what extent the time-sharing technique described in this paper is of more general use. As a computer the PDP-1 is characterized by high speed and relatively small memory. Its low cost means that it will not ordinarily have to be shared by a very large number of users. Suppose we wanted a time-sharing system based on core-drum swapping on another computer. Suppose that

n = number of simultaneous users
t = time for a memory cycle
m = number of words in user's memory that have to be swapped
r = response time
f = fraction of time taken by swaps;

then we have

$r = n\,t\,m(1 + \frac{1}{f})$ under the assumption that the drum keeps up with the core memory and that the read and write halves of the core memory cycle are used separately.

In the present case if we put $f = .25$, $n = 5$, $t = 5 \times 10^{-6}$ sec $m = 4000$ we get $r = .5$ sec. The difference between this result and the actual maximum response time of .85 sec. comes mainly from the fact that the present drum system swaps a word about every 8 microseconds instead of every five microseconds which in turn comes from using a standard 1800 rpm motor on a drum on which each track has 4096 bits around.

If we were to make a similar system for the IBM 7090 computer, we could have $n = 5$, $t = 2 \times 10^{-6}$, $m = 16{,}000$ (the memory of this machine really consists of 16384 72 bit words) $f = .25$ and would get

$r = .75$ sec

Thus, on account of its much larger memory the 7090 would have about the same relation between number of users and maximum response time as the PDP-1. This is less satisfactory because the expense of the larger machine requires it to serve many more users. Nevertheless, such a system would still be useful. If we make the more optimistic but reasonable assumptions that only $\frac{1}{5}$ of the users sitting at typewriters will be in run status at a given time and that a 3 second response time is tolerable, then the system could accommodate 100 typewriters which is economically quite reasonable. This would require a better drum system than is available connected so as to allow memory swaps at core speed.

REFERENCES

1. J. GILMORE—Lincoln Lab memo (out of print)
2. C. STRACHEY—Time-Sharing in Large Fast Computers in Proceedings of the International Conference on Information Processing, UNESCO, Paris 15-20 June 1959, UNESCO, Paris, 1960, pp. 336-341.
3. J. C. R. LICKLIDER—"Man Computer Symbiosis". *IRE Transactions on Human Factors In Electronics,* (March 1960).

4. F. J. CORBATO—1962 WJCC An Experimental Time-Sharing System, Fernando J. Corbato, Majorie Merwin-Daggett, Robert C. Daley, 1962 Spring Joint Computer Conference.
5. J. MCCARTHY—Time Sharing Computer Systems in Management and the Computer of the Future edited by Martin Greenberger, M.I.T. Press, 1962.
6. PDP-1 manual—Digital Equipment Corporation, Maynard, Mass.
7. S. BOILEN—User's Manual for the BBN Time-Sharing System—Bolt Beranek and Newman, 50 Moulton St., Cambridge, Mass.